檔案輸出與輸入

- 檔案輸出入之重要性
  將程式執行結果儲存下來
  避免重覆性人工輸入大量資料

- 以程式對檔案做輸出入之步驟
  開啟檔案
  讀/寫檔案
  關閉檔案

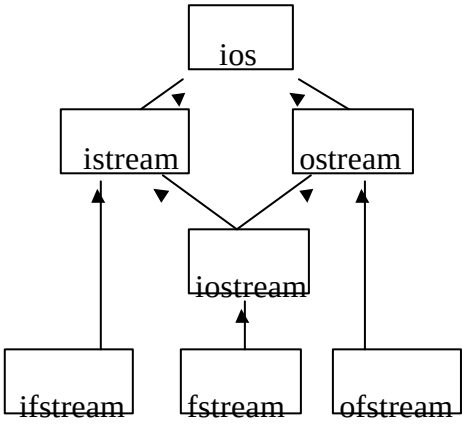
C/C++語言的檔案輸出入

	變數	輸入	輸出
C	FILE*	fscanf fgets fread	fprintf fputs fwrite
C++	ifstream ofstream	>>	<<

資料流(iostream)類別

- 資料流：輸出入設備
  - << 向左流入輸出入物件
  - >> 向右流入變數
- 資料流類別
  - ios: 輸出入的資料流
  - istream: 輸入的資料流
  - ostream: 輸出的資料流
  - ifstream: 支援檔案的輸入
  - ofstream: 支援檔案的輸出
  - fstream: 支援檔案的輸出入

C++之輸出入類別



標準資料流物件

- cin : 為istream 類別的物件，表示標準輸入
- cout : 為ostream 類別的物件，表示標準輸出
- cerr : 為ostream 類別的物件，表示標準錯誤輸出
- clog : 為ostream 類別的物件，表示有緩衝的標準錯誤輸出

ios的格式化旗標(flag)

- 在ios類別中有許多旗標(flag)可以設定，可用ios的類別成員函數setf()與unsetf()作設定，用ios::旗標名稱 來表該旗標

skipws	忽略輸入的空白鍵
left	輸出向左切齊
right	輸出向右切齊
dec	轉換成十進位制
oct	轉換成八進位制
hex	換成十六進位制
showbase	使用基底輸出
upercase	使用大寫
showpos	在正數前顯示+
scientific	科學記號輸出
fixed	浮點輸出採用固定小數點

ios格式化旗標的例子

```
#include  <iostream.h>
void main(){
  cout.setf(ios::hex);
  cout<< 100<<endl;    //輸出 64
  cout.setf(ios::showbase);
  cout << 100<<endl;   //輸出 0x64
  cout.setf(ios::uppercase);
  cout << 255<<endl;   //輸出 0XFF
  cout.unsetf(ios::hex);
  cout.setf(ios::showpos);
  cout<<100<<endl;     //輸出 +100
  cout.setf(ios::scientific);
  cout<<12345.6<<endl;
        //輸出 1.23456E+04
  cout.setf(ios::fixed);
  cout<<12345.6<<endl;
        //輸出 12345.6
```


ios運算子

ws	忽略輸入的空白
dec	轉換成十進制輸出入
oct	轉換成八進制輸出入
hex	轉換成十六進制輸出入
endl	換行
ends	送出字串結束符號
flush	將緩衝區資料送出
setw(int)	設定欄位寬度
setfill(int)	設定輸出的填入字元(預設為空白)
setprecision(int)	設定準確度的位數
setiosflags(long)	設定指定的旗標
resetiosflags(long)	清除指定的旗標

有參數的運算子使用前要先 #include <iomanip.h>

ios運算子之實例

```
#include  <iomanip.h>
void main(){
  cout<<setw(5)<<12<<endl;
  cout<<setw(15)<<"Computer"<<endl;
  cout<<setfill('*');
  cout<<setw(15)<<"Computer"<<endl;
  cout<<setprecision(3)<<3.1415926535<<endl;
  cout<<hex<<100<<endl;
  cout<<oct<<100<<endl;
}
```

輸出

```
   12
       Computer
*******Computer
3.142
64
144
```


ios函數

函　數	用　途
ch=fill()	傳回填入字元(預設是空白)
fill(ch)	設定填入字元
p=precision()	取得準確的位數
precision(p)	設定準確度
w=width()	取得目前的欄位寬
width(w)	設定目前的欄位寬
setf(flags)	設定指定的格式化旗標
unsetf(flags)	清除指定的格式化旗標
setf(flags)	設定格式化旗標

istream函式

函　式	用　途
>>	輸入基本資料型態
get(ch)	讀取一字元到 ch
getstr(str)	讀取一字串到 str
get(str,MAX)	讀取一字串到 str，最多 MAX 個字
get(str,MAX,DELIM)	讀取一字串到 str，最多 MAX 個字，或遇到 DELIM，DELIM 留在資料流
getline(str,MAX,DELIM)	讀取一字串到 str，最多 MAX 個字，或遇到 DELIM，DELIM 留在資料流
peek(ch)	讀取一個字元並保留在資料流
count=gcount()	傳回 get、getline、read 所讀的 byte 數
read(str,MAX)	讀入 MAX 個 byte 到 str 中
seekg()	設定檔案指位器到檔案開始處
seekg(pos,seek_dir)	設定檔案指位器
pos=tellg(pos)	傳回檔案指位器位置

11

ostream函式

函式	用途
<<	輸出基本資料型態
put(ch)	輸出字元 ch 到資料流
flush()	更新緩衝區
write(str,SIZE)	從陣列 str 輸出 SIZE 個 byte 到檔案
seekp(position)	設定檔案指位器的位置
seekp(position,seek_dir)	設定檔案指位器的位置
pos=tellp()	取得檔案指位器的位置

12

錯誤狀態位元

- 錯誤狀態位元負責報告輸入時發生的錯誤

位元名稱	發生時之狀況	位元值
goodbit	沒有錯誤(沒有任何位元被設定)	0x00
eofbit	抵達檔案尾端	0x01
failbit	運算失敗	0x02
badbit	不合法的操作	0x04
hardfail	硬體錯誤	0x08

錯誤位元函數

函數	說　明
int eof()	假使 EOF 位元被設定傳回真值
int fail()	假使 failbit、badbit 或 hardfail 位元被設定，傳回真值
int bad()	假使 badbit 或 hardfail 位元被設定，傳回真值
int good()	是否一切正常
clear(int)	清除資料流狀況

錯誤位元及其函數的使用

讀一整數資料並檢查

```
#include  <iostream.h>
void main(){
  int i;
  cout <<"i=";
  cin >> i;
  cout <<"Good ="<<cin.good()<<"\n";
  cout <<"Eof  ="<<cin.eof()<<"\n";
  cout <<"Fail ="<<cin.fail()<<"\n";
  cout <<"Bad ="<<cin.bad()<<endl;
 }
```

輸入

i=5

輸出

Good=1
Eof =0
Fail =0
Bad =0

輸入

i=a

輸出

Good=0
Eof =0
Fail =2
Bad =0

檔案的輸入/輸出類別

- **檔案輸出入類別**
  ifstream: **支援檔案的輸入**
  ofstream:**支援檔案的輸出**
  fstream :**支援檔案的輸出入**

- **檔案輸出入的步驟**
  Step 1: **開檔**
  Step 2: **讀寫檔案**
  Step 3: **關檔**

檔案讀寫的觀念

- **檔案是一種資料流**
- **檔案中有一個指位器指到下一次讀寫的位置。開檔時指位器指到檔案開始處**
- **檔案分文字檔與二進位檔**
- **在使用檔案輸出入時，程式中需加入** #include <fstream.h>

開檔的方法

- **方法一**: **用**ifstream**或** ofstream **的建構子**

```
ostream fout("data",ios::out);
```

檔名 **開檔模式**

- **方法二**: **用**open **類別成員函數**

```
ifstream fin;
fin.open("data",ios::in)
```

檔名 **開檔模式**

開檔模式

- 在開啟檔案使用open函數時可用底下模式

模式	結果
in	開檔做讀取
out	開檔做寫入
ate	從檔案結尾開始讀取或寫入 (AT End)
app	從檔案結尾開始寫入 (Append)
trunc	假使檔案存在，將其長度削為 0 (Truncate)
nocreate	假使檔案不存在，開啟錯誤
noreplace	假使檔案存在，開啟錯誤
binary	開啟二元模式檔案

資料流目前的位置

- 指定位置
  istream seekg(long 位置,位置算法)
  ostream seekp(long 位置,位置算法)

- 位置的算法
  ios::beg 由檔案開頭處算起
  ios::cur 由目前的位置算起
  ios::end 由檔案尾端往前算

- 取得資料流目前的位置
  long tellg()
  long tellp()

- 若檔案位置已至最後面，則eof為真

ifstream函數的實例

(取得檔案的長度)

```
#include  <iostream.h>
#include  <fstream.h>
void main(){
  char fname[20];
  cout<<"File name=";
  cin >>fname;     // 輸入檔名
  ifstream fin(fname,ios::in|ios::binary); //開檔
  fin.seekg(0L,ios::end);  //至檔案尾端
  cout<<"File length="<<fin.tellg()<<endl;
}
```


文字檔的讀取與寫入

- **文字檔的特性**
  檔案結束: 0x1a **即** ^Z
  換行 :
    在DOS**中** 0x0D 0x0A **即** ^M(CR) ^J(LF)
    在C/C++**中** 0x0A **即** ^J(LF)

- **不指明開檔模式即預設為文字檔**

- **文字檔的讀入**
  如同cin**一般使用，可用** >> **或** getline **或** get
- **文字檔的輸出**
  如同cout**一樣，可用**<< **或**put

文字檔的讀取與寫入之實例

檔案的複製

```
#include <iostream.h>
#include <fstream.h>
const int linelen=200;
void main(){
  char fname[20];
  ifstream fin;
  ofstream fout;
  char str[linelen];

  cerr<< "From file=";
  cin >> fname;
  fin.open(fname,ios::in);

cerr<< "To file=";
  cin >> fname;

fout.open(fname,ios::out);
  while (!fin.eof()){

fin.getline(str,linelen,'\n');
    fout <<str<<endl;
  }
}
```


二進位檔的輸入與輸出

- **二進位檔的特性**
  以資料的原始型態存檔
- **二進位檔的讀入**:
  使用ifstream**的**read**做輸入**
- **二進位檔的輸出**
  使用ofstream**的**write**做輸出**

輸出運算子的重載

- **標準的**cout**指令中的**<<**只能將基本型態做輸出**
- cout << obj

  可視為呼叫 operator << (cout,obj)

  可用

  ostream& operator<<(ostream &os, **類別名**& **參數名稱**)

輸出運算子重載的例子

複數的輸出

```
#include  <iostream.h>
class complex{
   float real;
   float imag;
 public:
   complex(float r=0, float i=0){
     real=r;   imag=i;  }
   friend ostream& operator <<(ostream& , complex&);
};
ostream& operator<<(ostream& os, complex& c){
 os << c.real <<"+ ";
 os << c.imag <<"i";
 return os;
}
```

```
void main(){
 complex a(2,-1);
 cout <<a;
}
```

輸出

2+ -1i

輸入運算子的重載

- **標準的**cin**指令中的**>>**只能將基本型態做輸入**
- cin>> obj

  可視為呼叫 operator >> (cin,obj)

  可用

  istream& operator>>(istream &is, **類別名**& **參數名稱**)

輸入運算子重載的例子

個人資料的輸入

```
#include <iostream.h>
class man{
  char name[40];
  char phone[10];
  public:
    friend istream&
        operator>>(istream&, man&);
    friend ostream&
        operator<<(ostream&, man&);
};
istream& operator>>(istream& is,
     man& m){
  cout <<"Name:";
  is  >> m.name;
  cout << "Phone:";
  is >> m.phone;
   return is;
}
ostream& operator<<(ostream& os,
          man &m){
  os << "Name : "<<m.name<<endl;
  os << "Phone: "<<m.phone<<endl;
  return os;
}
void main(){
  man Tom;
  cin >> Tom;
  cout << Tom;
}
```